Amendments to the Drawings

The attached sheet(s) of drawings includes changes to Fig(s) 3, 4, and 5. The sheet(s), which includes Fig(s) 1, 2, 3, 4, and 5, replaces the original sheet(s) including Fig(s) 1, 2, 3, 4, and 5.

Attachment: Replacement Sheet(s)

REMARKS

Claims 6-10, 16-20, 26-31, 33, 35-39, 41-45, 47 and 48 are pending in the application. Claims 6-10, 16-20, 26-31, 33, 35-39, 41-45, 47 and 48 stand rejected. Applicants herein amend claims 6, 8, 9, 16, 18, 19, 26, 28, 29, 31, 36-38, 43, 44, and 48. Further review and examination is respectfully requested.

Claim Rejections – 35 USC § 103

Claims 6-10, 16-20, 26-31, 35-38, 41-44, 47, and 48 stand rejected under 35 U.S.C. § 103(a) over U.S. Patent No. 6,735,598 to Srivastava in view of U.S. Patent No. 7,159,007 to Stawikowski, and U.S. Patent No. 7,376,755 to Pandya.

According to section 2141 of the MPEP, the Office may reject a claim as obvious in the instance that "the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains." MPEP, § 2141. Rejections based on obviousness cannot be sustained by conclusory statements. Rather, such conclusion must be supported "evidence which as a whole shows that the legal determination sought to be proved (i.e., the reference teachings establish a *prima facie* case of obviousness) is more probable than not." MPEP, § 2142. (Emphasis original).

Regarding claim 31, Applicants submit that the combination of Srivastava, Stawikowski, and Pandya fail to teach or suggest:

> Receiving, from a client, a query that invokes *intermediate language code*; [and]
>
> compiling by the runtime environment *during the runtime of the database management system*, the intermediate language code into an *expression encoded in the native query language*. (Emphasis added).

Applicants appreciate that this subject matter has not been examined as of yet; however, since this subject matter is similar to previously presented subject matter Applicants will address the applicability of the relevant sections of the cited art to the claim as amended.

The Office asserts the following n conjunction with the rejection of claim 31:

> line 66 to col. 4, line 3). Srivastava teaches the claimed, executing instructions from a memory in the database server (Fig. 11, col. 3, lines 20-26, the source code is compiled by a compiler in the database system to produce executable code and the compiler then modifies the table so that it relates the class and subclass specified in the package's name to the location of the code in the database system).
>
> Srivastava teaches analogous to the claimed, writing said application code as .NET managed code (Fig. 6, col. 10, lines 10-15, template 601 is written in the PL/SQL programming language. It may, however, be written in any language which the database system is able to compile). Srivastava does not explicitly teach .NET

Action, p. 3-4. Applicants respectfully submit that the cited sections of Srivastava fail to teach the excerpted portion of claim 1 above.

Applicants have claimed a process where intermediate language code is compiled during runtime into code that can be processed by the database management system. In addition, since the database management system uses the context of the user to determine whether an action can be taken and the client is accessing the database via intermediate language code (that is first compiled into an expression and then submitted to the database), the context of the user associated with the client is encapsulated and exposed to the database management system. In this regard, the expression can be submitted in accordance with the privileges associated with the client.

In contrast, col. 10, lines 10-15 of Srivastava seem to merely teach that user-defined classes can be written in PL/SQL. In particular, nothing in this text indicates that this code is compiled during the runtime operation of the database management system:

> In the preferred embodiment, template **601** is written in the PL/SQL programming language. It may, however, be written in any language which the database system is able to compile. For details on PL/SQL, see Scott Urman, Oracle8 PL/SQL Programming, ISBN: 0-07-882305-6, available from McGraw-Hill. Each method of the ORDSource sub-

Srivastava, col. 10, lines 10-15. A teaching or suggestion that intermediate language code is complied during runtime into an expression is absent from the cited text.

FIG. 11 and col. 3, lines 20-29 also fail to show the newly added subject matter. Rather, these sections of Srivastava seems to indicate that code is complied into the database and then it is invoked:

> Installing a Package: FIG. **11**
>
> At **610** in FIG. **6** is shown the SQL/PL statement that adds the following code for a package to the SOURCE module. The name following PACKAGE BODY is the package name, which has the form <ord class name>_<subclass name>_<module name>. For the package for the REALAUDIO subclass of the ORDX class in the module for the ORDSource type, PACKAGE BODY would be followed by ORDX_REALAUDIO_SOURCE. FIG. **11** shows how the database system responds to statement **610**. In system **1101** of FIG. **11**, compiler **1103** receives the package name, in this case, ORDX_REALAUDIO_SOURCE, at **1105**, and the package source code at **1107**. Compiler **1103** compiles source code **1107** to produce compiled code **1131**. Compiler **1103** also parses package name **1105** to obtain the name of the portion of code table **506** that contains the row for the package, here SOURCE **512**, and the name of the package, here REALAUDIO. With ORDSource classes, SOURCE represents the class and REALAUDIO the subclass. Compiler **1103** provides these names to name resolver **1119**, which responds by returning package row **1121** for the subclass, if the row exists. Compiler **1103** then uses blob handle **1133** from the row's column **510** to locate BLOB **1127** for the package. Thereupon, compiler **1103** writes compiled code **1131** to blob **1127**. If there is no row for the subclass **512**, the compiler inserts a new row **516** in code
>
> table **506**, with the subclass's name in column **508** and the handle for BLOB **1127** in column **510** and then writes compiled code **1131** to BLOB **1127**. The database system of the preferred embodiment also has provisions for executing compiled code contained in files external to the database system; thus, a package can also be implemented using any programming language for which there is a compiler on the computer system upon which the database system is executing.

Package Execution: FIG. **11**

> The database system of the preferred embodiment dynamically links the code for a method belonging to an ORDSource subclass at the time that code being executed by the database system invokes the method. Linking is done by name. The invocation of the method specifies the ORD-Source object, the method name, and the arguments for the invocation. The database system determines that the object is an ORDSource object and reads the object's subclass from type of source attribute **411**. As shown in FIG. **11**, the database system then provides the object's class **1111**, subclass **1113**, the method name **1115**, and the arguments to dynamic linker **1109**, which passes class name **1123** and subclass name **1125** to name resolver **1119**. Name resolver **1119** uses class name **1123** and subclass name **1125** to locate package row **516** for the subclass's package and returns the row as shown at **1121** to dynamic linker **1109**. Dynamic linker **1109** uses blob handle **1133** in row **516** to locate BLOB **1127** and then uses method name **1115** to locate method code **1129** in BLOB **1127**. Thereupon, dynamic linker **1109** invokes method code **1129** with arguments **1117**. The database system of the preferred embodiment maintains a cache in the main memory of the computer system upon which the database system is running, and once code in BLOB **1127** has been invoked, BLOB **1127** is copied to the cache, where it remains as long as it is being invoked with sufficient frequency that the cache management system does not remove it from the cache.

Srivastava, col. 10, line 42 through col. 11, line 39. According to these excerpted sections, it seems like the database obtains the package and complies it. After it is compiled into a blob, it is inserted into the database. Thereafter, methods of the blob can be invoked. Notably absent from this text is any indication that code in an intermediate language is invoked and complied during runtime into an expression that can be submitted to the database management system.

Further, it seems that Srivastava teaches a technique for organizing blobs so that their methods can be invokes. Claim 1, on the other hand, complies the intermediate language code into an expression, e.g., a search query, etc., that is submitted to the database. Simply put,

Srivastava seems to teach a technique for accessing methods stored in a database whereas claim 1 claims submitting expressions to a database.

The Office also assets that Stawikowski shows a technique for invoking .Net code in a database:

> code. However, Stawikowski teaches the claimed, invoking .NET managed code (col. 1, lines 62-63 and col. 2, lines 6-12, an applications server (J2EE, NET, etc.), a database management system (DBMS) server, the remote equipment comprises at least one processing unit, is capable of connecting to at least one item of automation equipment through an IP network and executing a program or a set of computer programs); Thus, it would have been obvious to one of ordinary skill in the data

Action, p. 4. The cited section of Stawikowski, however teaches something different than what has been alleged. Specifically, the cited section reads as follows:

> In the following, the term "remote equipment" may denote a personal computer, a portable telephone, or a PDA (Personal Digital Assistant) type equipment, or a computer server such as an ASP (Applications Service Provider), an applications server (J2EE, NET, etc.), a WEB server, a WAP server, a database management system (DBMS) server, an integrated management software (PGI) server, an ERP (Enterprise Resource Planning) server, an EAI (Enterprise Application Integration) server, an electronic document management (EDM) server, a business to business electronic shopping (B-to-B) server, a station for programming automation equipment, or any other computer system. Remote equipment may also be used to refer to a set of remote items of equipment communicating with each other. The remote equipment comprises at least one processing unit, is capable of connecting to at least one item of automation equipment through an IP network and executing a program or a set of computer programs. Some automation equipment such as dialog terminals may also be considered as remote equipment.

Stawikowski, col. 1, lines 60 through col. 2, lines 12. While the Office is correct in noting that the excerpted portion of Stawikowski includes the phrase .NET, the text is merely providing a definition of the term "remote equipment" and does describe invoking .NET code as claimed. In

particular, nothing in this section teaches or suggests that code in an intermediate language is invoked and complied during runtime into an expression that can be submitted to a database management system. Pandya was not cited as showing subject matter similar to the emphasized portions of claim 1 analyzed above. Consequently, since Srivastava and Stawikowski fail to teach or suggest the subject matter of claim 1, Srivastava in combination with Stawikowski and Pandya fail to render claim 31 *prima facie* obvious. Accordingly, Applicants respectfully request reconsideration of the rejection of claim 31 for these reasons.

Insomuch as claims 6-9 and 36 depend from claim 31 these claims define over the combination of Srivastava, Stawikowski, and Pandya for at least similar reasons as claim 31. Accordingly, Applicants respectfully request reconsideration of the rejections of these claims.

Turning to independent claim 37, it recites similar subject matter to claim 31, *mutatis mutandis*. Moreover, the Office has rejected this claim using the same rationale as claim 31. Consequently, Applicants respectfully submit that the combination of Srivastava, Stawikowski, and Pandya fails to render claim 37 obvious for at least similar reasons as claim 31. Accordingly, Applicants respectfully request reconsideration of the rejection of claim 37.

Insomuch as claims 16, 18, 19, 38, and 42 depend from claim 37 these claims define over the combination of Srivastava, Stawikowski, and Pandya for at least similar reasons as claim 37. Accordingly, Applicants respectfully request reconsideration of the rejections of these claims.

Turning to independent claim 43, it recites similar subject matter to claim 31, *mutatis mutandis*. Moreover, the Office has rejected this claim using the same rationale as claim 31. Consequently, Applicants respectfully submit that the combination of Srivastava, Stawikowski, and Pandya fails to render claim 43 obvious for at least similar reasons as claim 31. Accordingly, Applicants respectfully request reconsideration of the rejection of claim 43.

Insomuch as claims 26, 28, 44 and 48 depend from claim 43 these claims define over the combination of Srivastava, Stawikowski, and Pandya for at least similar reasons as claim 43. Accordingly, Applicants respectfully request reconsideration of the rejections of these claims.

CONCLUSION

Applicants request the Examiner reconsider the rejections and issue a Notice of Allowance of all the claims.

Date: May 24, 2011

/Michael J. Swope/
Michael J. Swope
Registration No. 38,041

Woodcock Washburn LLP
Cira Centre
2929 Arch Street, 12th Floor
Philadelphia, PA 19104-2891
Telephone: (215) 568-3100
Facsimile: (215) 568-3439